Comparing applicability of prevalent Clustering Algorithms for Document Clustering

Bachelor's Thesis submitted

to

Prof. Dr. Wolfgang K. Härdle

and

Prof. Dr. Nadja Klein

Humboldt-Universität zu Berlin
School of Business and Economics
Ladislaus von Bortkiewicz Chair of Statistics

by

Luisa Krawczyk
(573814)

in partial fulfillment of the requirements
for the degree of

Bachelor of Science in Business Administration

Paris, April 26, 2019

Abstract

In statistical data analysis, Clustering Algorithms are supposed to find groups of similar points in a set of objects. I present two algorithms, k-means and hierarchical clustering, and compare their applicability for clustering high-dimensional sparse data as often dealt with when clustering documents. To demonstrate possible cases of application, I provide clustering applications on an example data set (the Quantlet data set) in programming language Python. This paper's objective is to guide the reader from a casual understanding of basic statistical concepts, such as those typically acquired in undergraduate studies, to a general understanding of Clustering Algorithms, while focusing on Document Clustering applicability.

Keywords: Clustering Algorithms, k-means, Hierarchical Clustering, Document Clustering, Vector Space Model, Clustering Evaluation

Contents

List of Figures

1. Introduction

Clustering has widespread applications in business analytics and market research, bioinformatics, medicine and many more fields. In Customer segmentation, grouping clients based on their purchase history enables retailers to target them for specific campaigns. In medicine, clustering patients might be helpful to predict the probability of having a heart attack for each subgroup.

In document clustering, textual documents are clustered into groups of similar documents in terms of topics or keywords. Applications include web document clustering for search users, automatic document organization or topic extraction.

I present two well established algorithms often used in document clustering - k-means and hierarchical clustering - and apply them on an example data set which was provided to me by the *Ladislaus von Bortkiewicz Chair of Statistics*. The application part can be categorized as 'keyword' clustering, which however is similar to document clustering, when leaving out its text *preprocessing*. It therefore only deals with 'the second step' of document clustering, the clustering application on the *term-document matrix*.

This work aims at creating a theoretical foundation for understanding clustering algorithms and how they can be used to group documents. While detailed literature is available for most of the concepts mentioned in this paper, I want to contribute to the understanding of clustering applications by presenting all concepts essential to applying clustering algorithms on a data set, in particular a set of documents, alltogether with concepts of data preprocessing and choice of the right parameters and an appropriate distance measure.

This paper is organized as follows. Chapter 2 presents the two Clustering techniques that are to be compared, Partitional k-means and Hierarchical Agglomerative Clustering, in a general context. Chapter 3 introduces the vector-space model for document clustering and outlines basic *preprocessing* techniques. In chapter 4, I present application results of clusterings that I developed in Python and go into detail about practical questions as the choice of number of clusters and cluster performance evaluation. Chapter 5 contains a conclusion.

2. Overview Partitional and Hierarchical Clustering Techniques

2.1 Partitional Clustering : The k-means algorithm

Partitional Clustering Techniques have in common that the number of clusters k has to be determined by the user.

The most prominent partitional Clustering technique is the k-means algorithm.

First described by polish mathematician Hugo Steinhaus in 1951 [Florek, K., Lukaszewicz, J., Perkal, J., Steinhaus, Hugo, Zubrzycki, S., 1951], the k-means algorithm is a very simple and intuitive way to partition data. The term "k-means" was first used by [MacQueen et al., 1967]. In spite of the fact that it was proposed over 50 years ago and thousands of clustering algorithms have been published since then, k-means is still widely used [Jain, 2010]. Its goal is to minimize the distance between data points and their barycentre in each cluster. It is hence a so-called distance-based clustering algorithm.

The algorithm has the following four steps:

Given a set of n observations with d dimensions, so $X_1, ..., X_n$ with each $X_i \in \mathbb{R}^d$ arranged in a $d \times n$ matrix [1]

$$X_{d,n} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{d,1} & x_{d,2} & \cdots & x_{d,n} \end{pmatrix}$$

1. The user has to randomly assign k centroids (data points in the data space, which can but do not have to correspond to the observations). The k centroids $C_1, ..., C_k$ are thus each $d \times 1$ vectors. The centroid matrix is

$$C_{d,k} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,k} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,1} & c_{d,2} & \cdots & c_{d,k} \end{pmatrix}$$

In case it is possible to select logical centroids, the algorithm will work faster. [2]

[1] more on the representation of observations in the vector-space model in Section 3.1

[2] k-means++ is a variation of k-means choosing the initial centroids "cleverly"

2. For each data point X_i for $i = 1, ..., n$ we calculate the distance to each of the k centroids. For simplicity, let us assume the euclidean distance. [3]

$$d(X_i, X_j) = \sqrt{\sum_{m=1}^{d} (x_{m,i} - x_{m,j})^2}$$

For each data point those k distances are being compared and the smallest is chosen as centroid that X_i belongs to.

$$\forall i : L(i) = \arg\min_{l=1,...,k} \|X_i - c_l\|_2$$

$$\text{with } \forall l : C_l = [i | L(i) = l]$$

At the end of this step, we have a label vector $L(i)$ of length n that assigns to each X_i his centroid $c_l \in [|1, ..., k|]$ as well as a Size vector $S(l) = Card(C_l)$ of length k which tells the number of data points assigned to each cluster.

3. For each one of the k computed clusters, we calculate a new centroid as the mean, or barycentre, of all the data points which belong to it.

$$\forall l : C_l = \frac{1}{Card(C_l)} \sum_{i \in C_l} X_i$$

We define the sum of squared distances of iteration t to be

$$N(t) = \sum_{i=1}^{n} \sum_{l=1}^{k} \|X_i^{(l)} - C_l\|_2^2$$

where $X_i^{(l)}$ denotes all X_i that belong to cluster C_l and $\|.\|_2^2$ is the squared Euclidean distance.

4. Repeat step 2 and 3 until the reduction in sum of squared distances is smaller than a certain ϵ i.e. until

$$t \quad \text{such that} \quad N(t-1) - N(t) < \epsilon.$$

K-Means can easily be computed with the help of *Scikit-learn*, a free software machine learning library for Python programming language. However, this [manual k-means] algorithm might be helpful to understand what is happening behind the scenes.

[3] Different distance measures can be employed here, we will see more about them in Section 3.3.

2.2 Hierarchical Clustering

Hierarchical Clustering techniques are connectivity-based. In contrast to partitional techniques, they do not require a predefined value for the number of clusters (k) but provide a tree of different cluster divisions, called *dendrogram* [4], for any number of clusters between 1 and n.

We distinguish two types of hierarchical Clustering algorithms:

Agglomerative Clustering which starts with n clusters (i.e. every data point has its own cluster) and then merges the closest pair of clusters at each step.

Divisive Clustering which starts with one all-inclusive cluster and then splits the groups with the largest distance at each step.

Agglomerative techniques are more common and these are the ones that I will compare to k-means.

The algorithm of an **Agglomerative Clustering** is composed of the following steps:

1. Calculate the distance matrix

$$D_{n,n} = \begin{pmatrix} d(X_1, X_1) & d(X_1, X_2) & \cdots & d(X_1, X_n) \\ d(X_2, X_1) & d(X_2, X_2) & \cdots & d(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ d(X_n, X_1) & d(X_n, X_2) & \cdots & d(X_n, X_n) \end{pmatrix}$$

where $d(X_i, X_j) = \sqrt{\sum_{m=1}^{d} (x_{m,i} - x_{m,j})^2}$ is the euclidean distance between X_i and X_j. [5]

D is a symmetric matrix with the diagonal values being 0. Now we define each data point as being one cluster, we thus start having n clusters.

2. Identify the most similar i.e. closest two clusters (r) and (s)

For now as one cluster is equivalent to one data point:	Later with (i) and (j) being clusters:
$d(X_r, X_s) = \min_{\substack{i,j \in [\|1,\ldots,n\|] \\ i \neq j}} d(X_i, X_j)$	$d((r),(s)) = \min_{\substack{i,j \in [\|1,\ldots,n\|] \\ i \neq j}} d((i),(j))$

which corresponds to finding the smallest values in our distance matrix.

3. Merge clusters (r) and (s) into a single cluster (rs) and update the distance matrix by deleting the row and column corresponding to X_r and X_s as well as adding a row and column corresponding to the newly formed cluster (rs).

[4] An example of a dendrogram is Figure 4.5 shown in Section 4.2

[5] Of course, any other distance measure can be used here.

$$D_{n-1,n-1} = \begin{pmatrix} \begin{pmatrix} d(X_1,X_1) & \cdots & d(X_1,X_{r-1}) & d(X_1,X_{r+1}) \cdots & d(X_1,X_{s-1}) & d(X_1,X_{s+1}) & \cdots & d(X_1,X_n) & d(X_1,(rs)) \\ d(X_2,X_1) & \cdots & d(X_2,X_{r-1}) & d(X_2,X_{r+1}) \cdots & d(X_2,X_{s-1}) & d(X_2,X_{s+1}) & \cdots & d(X_2,X_n) & d(X_2,(rs)) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ d(X_n,X_1) & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & d(X_n,X_n) & d(X_n,(rs)) \\ d((rs),X_1) & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & d((rs),X_n) & d(X_{rs},(rs)) \end{pmatrix} \end{pmatrix}$$

where $d((rs), X_1)$ can be calculated using different linkage criteria, for example $d((rs), X_1) = \min_{r,s}[d(X_i, X_r), d(X_i, X_s)]$ for single linkage. [6]

4. Repeat from step 2 unless all objects are in one cluster, then stop.

In contrast to partitional techniques, where data points can change cluster affiliation from iteration to iteration, in hierarchical clustering each affiliation decision is definitive; data points that belong to a child cluster also belong to the parent cluster.

Linkage criteria

The key element of this algorithm is its dissimilarity measure, also referred to as proximity measure, which consists of two things: a distance metric between pairs of observations [7] and a linkage criterion which stipulates the dissimilarity/ proximity of two sets as a function of the pairwise distances of observations in the sets.

The most prominent linkage criteria:

Simple Linkage	$d(A,B) = \min_{X_i \in A, X_j \in B} d(X_i, X_j)$
Complete Linkage	$d(A,B) = \max_{X_i \in A, X_j \in B} d(X_i, X_j)$
Average Linkage	$d(A,B) = \frac{1}{\lvert A\rvert \cdot \lvert B\rvert} \sum_{X_i \in A} \sum_{X_j \in B} d(X_i, X_j)$ where $\lvert A\rvert = Card(A)$
Centroid Linkage	$d(A,B) = d(c_A, c_B)$ where c_A is A's centroid
Ward's method	$d(A,B) = \frac{d(c_A,c_B)}{\frac{1}{\lvert A\rvert} \cdot \frac{1}{\lvert B\rvert}}$ which corresponds to the variance increase when merging the clusters.

When using hierarchical clustering, some linkage criteria can be more suitable than others, i.e. achieve a segmentation where similarity between points in one group as well as dissimilarity between groups is greater.

[6] More about this in the following section.

[7] for now said to be the Euclidean distance, more on distance measures in Section 3.3

3. Document Clustering

This chapter intends to clarify the elements of Document Clustering and situate it within the Data Science context. It describes how documents are transformed to a numerical matrix prior to clustering.

3.1 Concepts of Information Retrieval

3.1.1 Data preprocessing

The preprocessing techniques used prior to Document Clustering are methods of Information Retrieval. They are common in Text Mining applications, which deal with unstructured or semistructured data and serve to quantify the nature of a textual document. The tools used are referred to as Natural Language Processing.

Those preprocessing techniques include

- Tokenization: the process of parsing text data into smaller units (tokens) such as words and phrases,
- eliminating stop words: words like "the", "and", "or" are not very helpful in revealing the essential content of a document,
- recognizing stems of words:"cluster" and "clustering" carry very similar information and should thus be treated as the same word
- distinguishing important words from unimportant words: e.g. "according" reveals much less information about the content of an Econometrics course document than "heteroscedasticity",
- dealing with synonyms (different words with the same meaning) and homonyms (words with the same spelling but with distinct meanings) etc.

Only after this often extensive preprocessing, it is possible to obtain "descriptors" i.e. sets of words that describe the content of each document.

3.1.2 Vector-space model

Documents are represented using the vector-space model. The descriptors obtained after preprocessing are then arranged in a so-called *Term-Document-Matrix* or extensions of it (e.g. TF-IDF for term frequency-inverse document frequency).

Definition Term-document matrix [Karypis et al., 2000]
The term-document matrix is defined as: $(f_{i,j})_{\substack{i=1,\dots,n \\ j=1,\dots,d}}$ where $f_{i,j}$ is the frequency with which term i appears in document j, n is the vocabulary size and d is the number of documents. Each column of the matrix corresponds to the j^{th} document vector.

An example is given in Table 4.1 of the next chapter.

As this paper deals with the "second step" of document clustering, I do not further detail any extensions or Text Mining Preprocessing techniques. To gain a deeper understanding of those, see [Baeza-Yates et al., 2011].

3.2 Document Clustering Techniques

Based on the term-document matrix described above, documents can finally be clustered into groups of similar documents. Documents in one group will have more terms in common with each other than with documents in other groups.

An important feature of document clustering as opposed to other clustering applications is the number of variables that often exceeds many thousand. Furthermore, term-document matrices can be very sparse, i.e. contain many zeros. Applying clustering algorithms to high-dimensional sparse data is a challenge tackled by the choice of an appropriate document clustering technique.

The most common techniques used are k-means and hierarchical agglomerative clustering, which is why I will compare those two in the application part. The numerous other algorithms introduced include density-based algorithms as DBSCAN, Spectral Clustering or hybrid models that combine features from both partitional and agglomerative approaches as in [Zhao et al., 2005]. Beyond that there are numerous novel algorithms presented by researchers as [Joel Larocca Neto and Alexandre D. Santos and Celso A.A. Kaestner and Neto Alexandre and D. Santos and Celso A. A and Kaestner Alex and Alex A. Freitas and Catolica Parana, 2000] who present the Autoclass Algorithm for clustering and summarizing documents.

3.3 Distance measures

The distance between two documents in the vector-space model can be calculated in different ways. The choice of the right distance or similarity measure can be crucial in clustering applications. There is no "correct" measure, it always depends on the data set.

NB.: A similarity measure is simply the 'opposite' of the distance measure.

3.3.1 Euclidean distance

The most intuitive distance measure is often thought to be the Euclidean norm, or L_2-Norm, which is the shortest line between two data points in the vector-space model. For two data

points X_i and X_j the Euclidean distance is defined as

$$d_E(X_i, X_j) = \|X_i - X_j\|_2 = \sqrt{\sum_{m=1}^{d}(x_{m,i} - x_{m,j})^2}$$

3.3.2 Manhattan distance

The Manhattan distance is in fact the L_1-Norm defined as

$$d_M(X_i, X_j) = \|X_i - X_j\|_1 = \sum_{m=1}^{d}|x_{m,i} - x_{m,j}|$$

3.3.3 Cosine similarity

Introduced by [Salton et al., 1975], the cosine similarity is often thought to be more suitable for document clustering as the length of the vector, i.e. the number of terms in a document, does not impact the distance measure, but the relative distribution of terms will matter. The cosine similarity is defined as

$$s_C(X_i, X_j) = \frac{X_i \times X_j}{\|X_i\|_2 \|X_j\|_2}$$

where $\times$ denotes vector multiplication and $\|\|_2$ is the L_2-Norm.

The Cosine distance accordingly is defined as

$$d_C(X_i, X_j) = 1 - s_C(X_i, X_j)$$

3.3.4 Jaccard Coefficient

The Jaccard Coefficient is similar to the Cosine similarity. It compares the weight of terms documents share with the weight of their overall terms. The Jaccard similarity is defined as

$$d_J(X_i, X_j) = \frac{X_i \times X_j}{\|X_i\|_2^2 + \|X_j\|_2^2 - X_i \times X_j}$$

More on distance measures can be found in [Strehl et al., 2000] and [Huang, 2008] who have experimented with different measures and discussed their effectiveness in text document clustering. The general opinion is Cosine similarity and the Jaccard Coefficient are most suitable for Document Clustering.

4. Application

In this chapter I will present my results when using k-means and hierarchical Clustering on the Quantlet data set.

4.1 The data set

The documents I cluster are codes and programs produced by researchers at the *Ladislaus von Bortkiewicz Chair of Statistics* of Humboldt University Berlin. Those codes and programs, called Quantlets, are published on the Chair's GitHub page, see GitHub.com/Quantlet. Quantlets are accompanied by a metainfo file, which is a text file containing a name of the Quantlet, the name of the author, a description, some keywords and where it was published.

When translating into a text mining environment, you could say that the most relevant words from each document have been carved out by the author stating his keywords. For my Clustering I thus use the (on average) 9 keywords per document which provide a term-document matrix:

document	basis	graphical	orthogonal	plot	probability
ADM/HermPolyPlot/Metainfo.txt	1	2	1	1	1
ARR/ARRboxage/Metainfo.txt	0	1	0	1	0
ARR/ARRboxgscit/Metainfo.txt	0	1	0	1	0
ARR/ARRboxhb/Metainfo.txt	0	1	0	1	0
ARR/ARRcormer/Metainfo.txt	0	1	0	1	0

Table 4.1: "Head" of the quantlet data set; first five terms for the first five documents

The resulting document clusters are supposed to share more keywords with those documents in their own cluster than with documents in other clusters.

The Quantlet term-document-matrix is one of size 2064×1286, $d = 1286$ terms (i.e. variables) and $n = 2064$ documents.

4.2 Specific properties of the Quantlet data set

The Quantlet term-document matrix has some specific properties different from usual document clustering problems.

1. Not as many terms per document as in the case of extraction by preprocessing text data. Rarely more than 1 same term per document so that computing variations of the term-document matrix, as for example a TF-IDF matrix, becomes redundant.

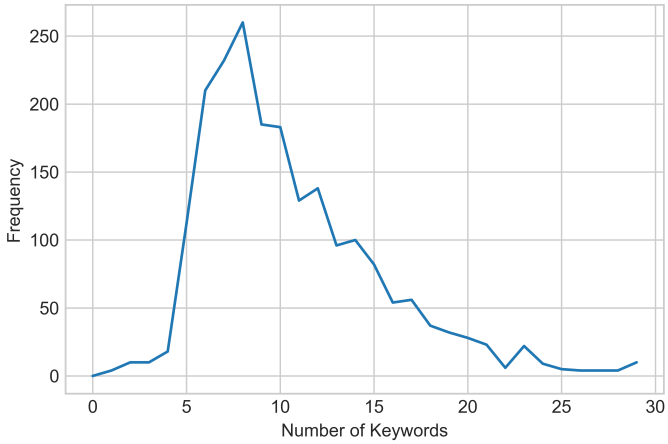


Figure 4.1: Number of Keywords Distribution
NumberOfKeywords

2. Unequally distributed number of terms per document as each author had a different understanding of how many keywords were appropriate. As visible in Figure 4.1, most documents have between 5 and 15 keywords (88 % of the documents to be exact). In document clustering this would correspond to different lengths of each document.

Implications

Doubts concerning the applicability of the Euclidean distance can be dropped as the Quantlet term-document matrix is nearly a binary matrix with few entries different than 1. That is why I did not apply other similarity measures, as the cosine similarity or the Jaccard coefficient. Comparing their performance could still be content for further analyses.

4.3 k-means

The k-means algorithm runs at a computationally very low cost which generally makes it the choice number one clustering method. K-means performed quite well on the Quantlet data set, finding *relatively* equally sized clusters making sense at the first glance.

Table 4.2 shows three of ten clusters that I generated using my [manual k-means algorithm] with Euclidean distance and random centroid initialization. As our observations i.e. file names do not reveal information about the nature of its content, I decided to look at the final centroid values (after 7 iterations). The length d centroid vector has most of its elements equal to or close to zero; namely for the terms which do not appear in any of its cluster's documents. The highest centroid vector values identify the terms appearing in most of the documents' keywords. For example, the term "option" appears nearly 2 times on average in every document of Cluster 1, which I labelled being a "Finance Cluster" given the prevalent terms of its documents. The

Cluster 1:	"Finance"	Cluster 2: "Visualization	of data"	Cluster 3:	"Time series"
option	1.96	visualization	1.11	plot	0.24
price	1.22	plot	0.93	time	0.17
financial	0.59	graphical	0.91	series	0.15
black	0.55	representation	0.91	model	0.12
scholes	0.54	data	0.77	simulation	0.11

Table 4.2: For 3 of 10 clusters computed with k-means: The terms having highest centroid-values and their shortened centroid-values
[k-means centroids]

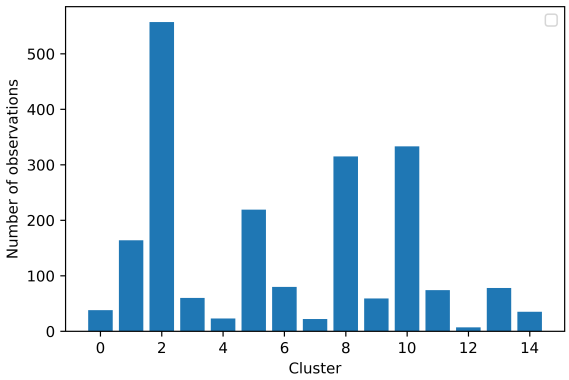


Figure 4.2: Number of observations per cluster when using k-means with k=15
[Cluster sizes]

generally lower centroid values in Cluster 3 are partly due to the higher number of observations contained in that cluster.

4.3.1 Determining the optimal number of clusters

The most often cited disadvantage of k-means as opposed to hierarchical clustering techniques is the necessity of choosing the number of clusters k beforehand. However, there are different methods to solve this problem.

Elbow Method

One of those is the *Elbow Method* which consists in comparing the sum of within-cluster squared distances of a clustering for different values of k. When there is a steep decrease of sum of squared distances from $k = \tilde{k} - 1$ to $k = \tilde{k}$, then $\tilde{k}$ is the optimal number of clusters.

I recall that the total sum of squared distances between data points and their cluster's barycentre is

$$\sum_{i=1}^{n} \sum_{j=1}^{k} \|X_i^{(j)} - c_j\|_2^2$$

where c_j is the cluster j , $X_i^{(j)}$ all X_i that belong to cluster j and $\|.\|_2^2$ the squared Euclidean distance.

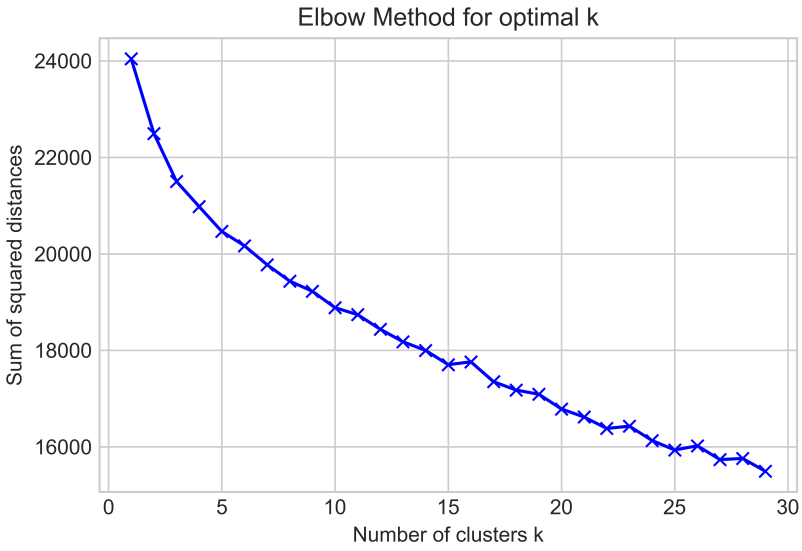


Figure 4.3: Sum of squared distances as a function of the number of clusters [sum of squared distances]

However, this approach does not always yield an obvious and unique number of clusters. Figure 4.3 shows the total sum of squared distances as a function of the number of clusters for the Quantlet Clustering. We cannot clearly identify an elbow, though $k = 3, k = 15, k = 22, k = 25$ are possible candidates. The choice of number of clusters will depend on the purpose and the assessment of the user.

Silhouette Score

Another method is checking the Silhouette Score (or other performance scores) for different values of k and choose one value with a high score. [1]

4.3.2 Centroid initialization problem

A more serious problem than determining the adequate number of clusters lies within the initialization of centroids. When running the hand-written algorithm described in section 2.1, each run with a different random initialization returns different clusters. Possible solutions to this problem are:

Consensus Clustering which consists of generating a set of clusterings from the same data set - using different techniques or the same algorithm run with different initializations - and combining them into a final clustering, where each data point's cluster affiliation is decided

[1] More about the Silhouette score in section 4.5 Evaluation

upon by majority vote. The goal of this combination process is to improve the quality of individual data clusterings. [VEGA-PONS and RUIZ-SHULCLOPER, 2011]

k-means++ which is an extension of classical k-means proposed by [Arthur, 2007]. It contains a seeding method for "cleverly" choosing the first centroids. The initialization of k-means++ works as follows:

1. One data point is chosen randomly to be a cluster centroid c_1. Let $\hat{k}$ be the initialization step. For now, $\hat{k} = 1$.

2. We compute $d(X_i) = d(X_i, c_1) \quad \forall i = 1, ..., n$

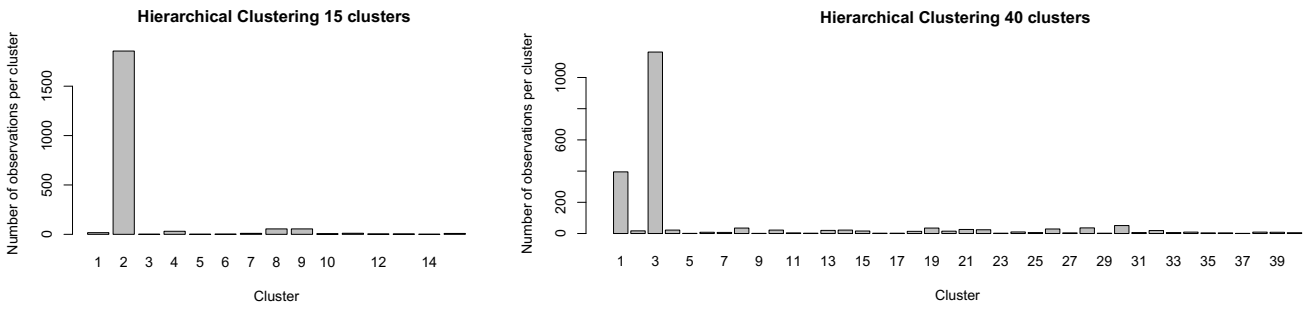
3. Choose another data point to be a new cluster centroid c_2 where the probability that X_i is chosen is proportional to $d(X_i)^2$. By definition, data points far away from c_1 are more likely to be chosen. Now we have $\hat{k} = 2$.

4. Repeat steps 2 and 3 while now computing $d(X_i) = \min_{c=c_1,...,c_{\hat{k}}} d(X_i, c)$ as the distance of X_i to its closest already computed centroid until $\hat{k} = k$ and k centroids have been chosen.

5. Proceed with standard k-means.

4.4 Hierarchical Agglomerative Clustering

Applying standard agglomerative hierarchical clustering on the Quantlet data did not yield satisfying results as the algorithm produced one big cluster containing most of the observations while allocating all remaining clusters only few observations.



(a) 15 clusters

(b) 40 clusters

Figure 4.4: Number of observations per cluster for a Hierarchical Clustering using Complete Linkage

[*Hierarchical Cluster sizes*]

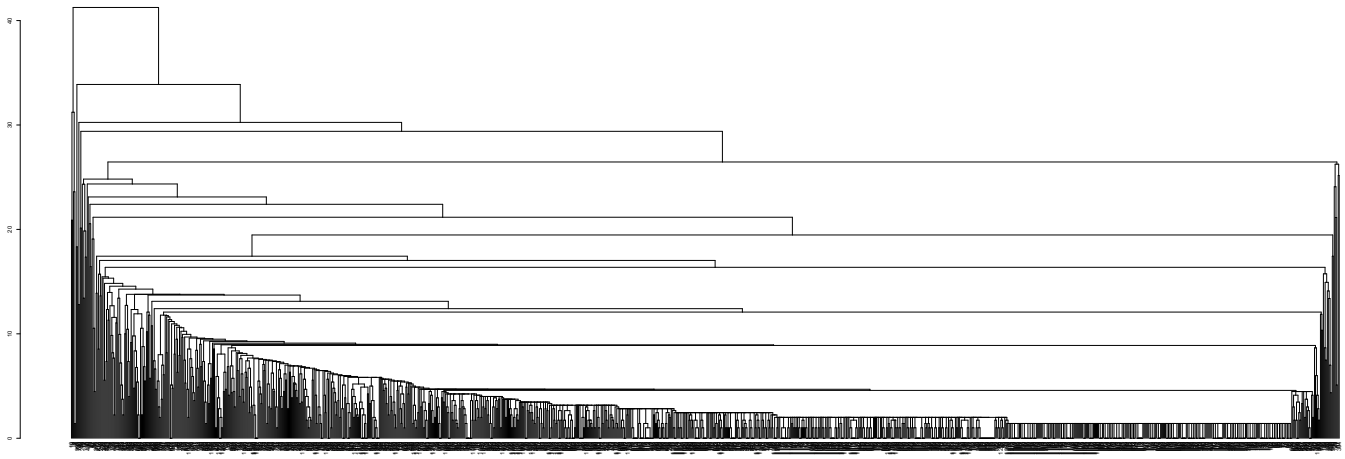


Figure 4.5: Dendrogram as a result of an agglomerative hierarchical Clustering using Complete Linkage for the Quantlet data set computed in R
[whole dendrogram]

Figure 4.4 shows the allocation of observations among clusters for an Agglomerative Hierarchical Clustering using Complete Linkage and 15 or respectively 40 clusters. As the aim of our clustering is to represent Quantlets in coherent groups, this repartition is not helpful. Now supposing those remote observations are outliers, we can remove them and cluster again. Nearly equivalent to this approach, is to choose a higher number of clusters and exclude those with few observations; though this method is even better for possibly joining observations that we would have classified as outliers. But even when choosing 40 clusters, we remain with a 1441-observation cluster and one 139-observation-cluster, while the other 38 clusters have less than 51 observations.

In a dendrogram, the y-axis marks the distance at which clusters merge, while the observations are placed along the x-axis for reasons of clarity. The dendrogram of Figure 4.5 shows that the first split results in 4 observations being put in one cluster and 2060 in the other. Even the fourth split leaves only 9 observations out of the large all-inclusive cluster. Cutting the dendrogram at 60 clusters still leaves 1028, so half of the documents, in one single cluster. In fact, we cannot find a satisfying split, even when ignoring outliers. The dendrogram can be represented in a truncated form; the user can either decide on a truncation distance or a maximum number of clusters to be displayed. An example code in R is provided for [hierarchical clustering truncation].

The phenomenon of unequal cluster sizes even intensifies when choosing single or average linkage instead of complete linkage. This is due to complete linkage generally producing rather equally sized clusters.

4.5 Clustering Performance Evaluation

I already evaluated the k-means algorithm "manually" by examining the cluster centers and checking for a common topic for each cluster. An "indirect" evaluation of the two clusterings concerns their utility in their intended application. As the goal of clustering our Quantlets is to represent them in coherent groups on the GitHub web site, clusters with more or less equal sizes are a lot more appropriate. This is why the manual and indirect evaluation suggests that the k-means algorithm performs better than hierarchical clustering.

In order to evaluate the quality of a clustering quantitatively, it would be best to have *true* labels for our documents, i.e. categories they belong to, as "portfolio optimisation", "risk theory" or "Clustering". This form of "external" evaluation would leave no doubt about which clustering techniques lead to the most correct result.

Unfortunately, real-world data, often do not provide such *true* labels. The absence of category information distinguishes data clustering **(unsupervised learning)** from classification or discriminant analysis **(supervised learning)**. The aim of clustering is to find structure in data and it is therefore exploratory in nature. [Jain, 2010]

The Quantlet data set does not provide the above-mentioned labels. Therefore, they require "internal" evaluation, where the clustering is summarized to a single quality score. I will present two such indices; the Silhouette Score and the Calinski-Harabaz Score.

4.5.1 Silhoutte Score

The Silhouette Score is a measure of how similar an observation is to its own cluster in comparison to other clusters. It was introduced by [Rousseeuw, 1987].

The *Silhouette value* of a data point i which was attributed to Cluster A is

$$s(i) = \begin{cases} \dfrac{b(i) - a(i)}{\max\{a(i), b(i)\}} & \text{if } Card(A) > 1 \\ 0 & \text{if } Card(A) = 1 \end{cases}$$

$$\text{where} \quad a(i) = \frac{1}{Card(A) - 1} \sum_{j \in A, i \neq j} d(i,j) \quad \text{is the average within-cluster dissimilarity}$$

$$\text{and} \quad b(i) = \min_{C \neq A} \sum_{j \in C} d(i,j) \quad \text{the average dissimilarity to the next closest cluster.}$$

Therefore we have $s \in [-1, 1]$ where $s = 1$ represents a very dense and separable clustering, $s = 0$ overlapping clusters and $s < 0$ an incorrect clustering. The higher the score, the better is our clustering performance.

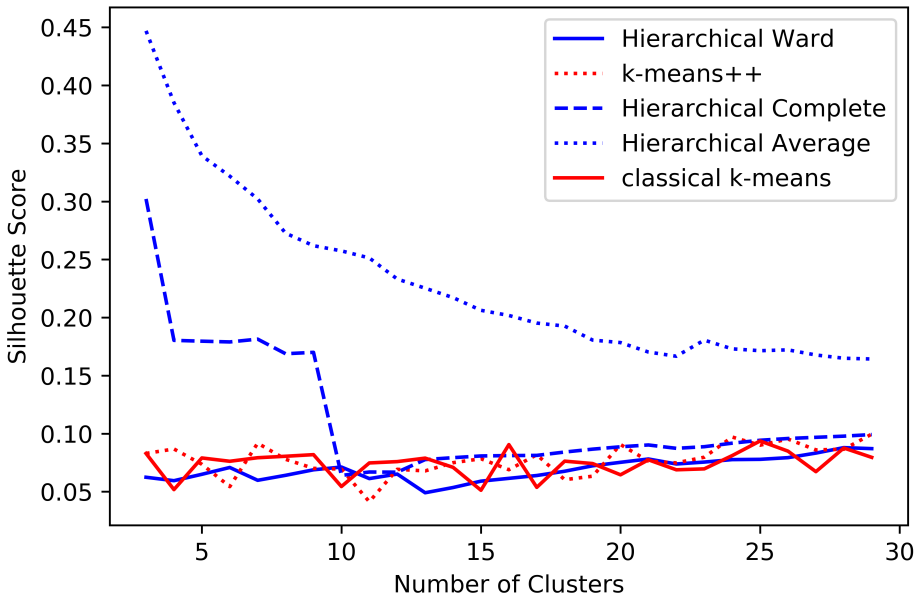


Figure 4.6: Silhouette Score for k-means and Hierarchical Clusterings [SilhouetteScore]

As we can infer from Figure 4.6, the Hierarchical Clustering with Average Linkage performed best in terms of Silhouette Score. Second best is the Hierarchical Clustering with Complete Linkage, while especially performing well for small values of k. K-means, K-means++ as well as the Hierarchical Clustering with Ward Linkage seem to perform worse.

Keeping in mind the very unequal distribution of observations among clusters for the Hierarchical Clustering with Average Linkage, the Silhouette Score seems inappropriate in choosing the "correct" Clustering technique.

4.5.2 Calinski-Harabaz Score

The Calinski-Harabaz Score, also known as Variance Ratio Criterion, is the ratio of the mean between cluster dispersion (i.e. cluster variance) and the within-cluster dispersion. [Caliski and Harabasz, 1974] The score s is defined per clustering.

Let k be the number of clusters. Then

$$s(k) = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{N-k}{k-1}$$

where N is the number of observations in the data set and Tr() the trace of matrix B_k and W_k respectively.

The within-cluster dispersion matrix is $W_k = \sum_{j=1}^{k} \sum_{X_i \in C_j} (X_i - c_j)(X_i - c_j)^T$

and the between-cluster dispersion matrix is $B_k = \sum_j n_j (c_j - c)(c_j - c)^T$

where C_j denotes the set of data points within cluster j , c_j its centroid vector, $n_j = Card(C_j)$ and c is the barycentre of the whole data set.

The score is lower bounded by 0 but has no upper bound. Higher scores indicate denseness and good separability of the clusters.

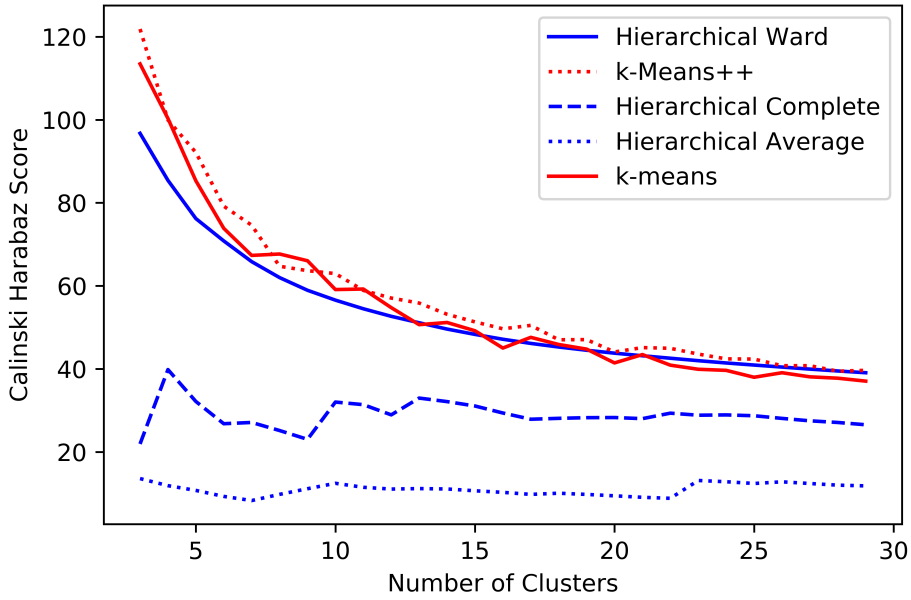


Figure 4.7: Calinski-Harabaz Score for k-means and Hierarchical Clustering [CalinskiHarabazScore]

Figure 4.7 shows the Calinski Harabaz Score for K-means and three different linkage criteria of the Hierarchical Clustering. Interestingly, the results of this analysis are the exact opposite of those proposed by the Silhouette Score. K-means and Hierarchical Ward Linkage perform best while Complete and Average Linkage perform worse. We also observe that the score generally decreases with the number of clusters.

4.6 Results

For this specific data set it seems more adequate to rely on a manual and indirect evaluation as the two quantitative scores presented are contradictory and also the hierarchical clustering does not satisfy the intended application. The k-means algorithm appears more appropriate.

I also found that for hierarchical clustering, while finding very unequal cluster sizes for the Quantlet data set, complete linkage is still rather computing more equal cluster sizes than single or average linkage.

5. Conclusion

Clustering is a powerful tool for understanding the structure of a data set, in particular a set of documents. In the previous sections I presented two commonly used clustering algorithms, k-means and agglomerative hierarchical Clustering, and introduced concepts crucial to the understanding of a document's numerical representation in the vector-space model.

I presented an application of both, k-means and agglomerative hierarchical clustering, on the Quantlet data set and discussed their performance on two different levels; a purely manual and indirect evaluation prefers the k-means algorithm over any kind of hierarchical clustering as it returns clusters with more or less even cluster sizes while a purely quantitative internal evaluation suggests different algorithms and linkage methods depending on the score used. While the application is similar to document clustering, it should rather be labeled "keyword clustering" as the term-document matrix is built by keywords instead of term frequencies. As part of the application of k-means, I presented some numerical approaches of determining the number of clusters and discussed the initialization problem.

Summarizing literature about document clustering, most find that beyond having relatively low computational requirements, partitional techniques are better suited for document clustering than hierarchical ones. [Aggarwal et al., 1999]

References

Aggarwal, C. C., Gates, S. C., and Yu, P. S. (1999). On the merits of building categorization systems by supervised clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 352–356. ACM.

Arthur, D.; Vassilvitskii, S. (2007). "k-means++: the advantages of careful seeding". In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Baeza-Yates, R., Ribeiro, B. d. A. N., et al. (2011). *Modern information retrieval*. New York: ACM Press; Harlow, England: Addison-Wesley,.

Caliski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

Florek, K., Lukaszewicz, J., Perkal, J., Steinhaus, Hugo, Zubrzycki, S. (1951). Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicum*, 2(3-4):282–285. `http://eudml.org/doc/209969`.

Huang, A. (2008). Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

Joel Larocca Neto and Alexandre D. Santos and Celso A.A. Kaestner and Neto Alexandre and D. Santos and Celso A. A and Kaestner Alex and Alex A. Freitas and Catolica Parana (2000). Document clustering and text summarization.

Karypis, M. S. G., Kumar, V., and Steinbach, M. (2000). A comparison of document clustering techniques. In *TextMining Workshop at KDD2000 (May 2000)*.

Kowalski, G. J. (2007). *Information retrieval systems: theory and implementation*, volume 1. Springer.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Strehl, A., Ghosh, J., and Mooney, R. (2000). Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)*, volume 58, page 64.

VEGA-PONS, S. and RUIZ-SHULCLOPER, J. (2011). A survey of clustering ensemble algorithms. In *International Journal of Pattern Recognition and Artificial Intelligence*, volume 25, pages 337–372. Advanced Technologies Application Center, 7A 21812, Siboney, Havana 12200, Cuba, World Scientific Publishing Company. `https://doi.org/10.1142/S0218001411008683`.

Zhao, Y., Karypis, G., and Fayyad, U. (2005). Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery*, 10(2):141–168.

Statutory Declaration

I declare that I have developed and written the enclosed Bachelors Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Bachelors Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere. I understand that violations of these principles will result in proceedings regarding deception or attempted deception.

Luisa Krawczyk

Paris, April 23, 2019